

09766402.07604

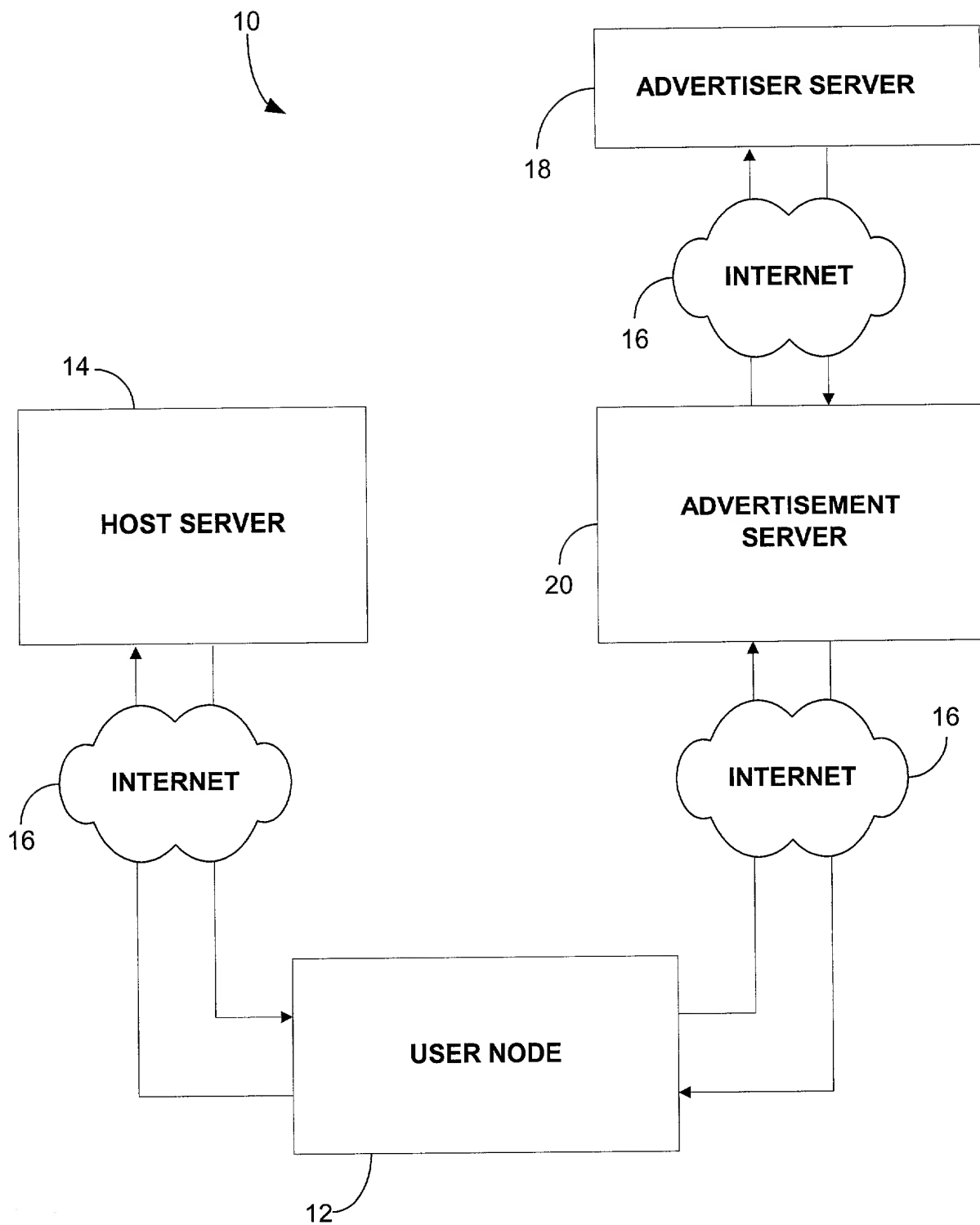


Figure 1

0976402-070601
T09070" 20429260

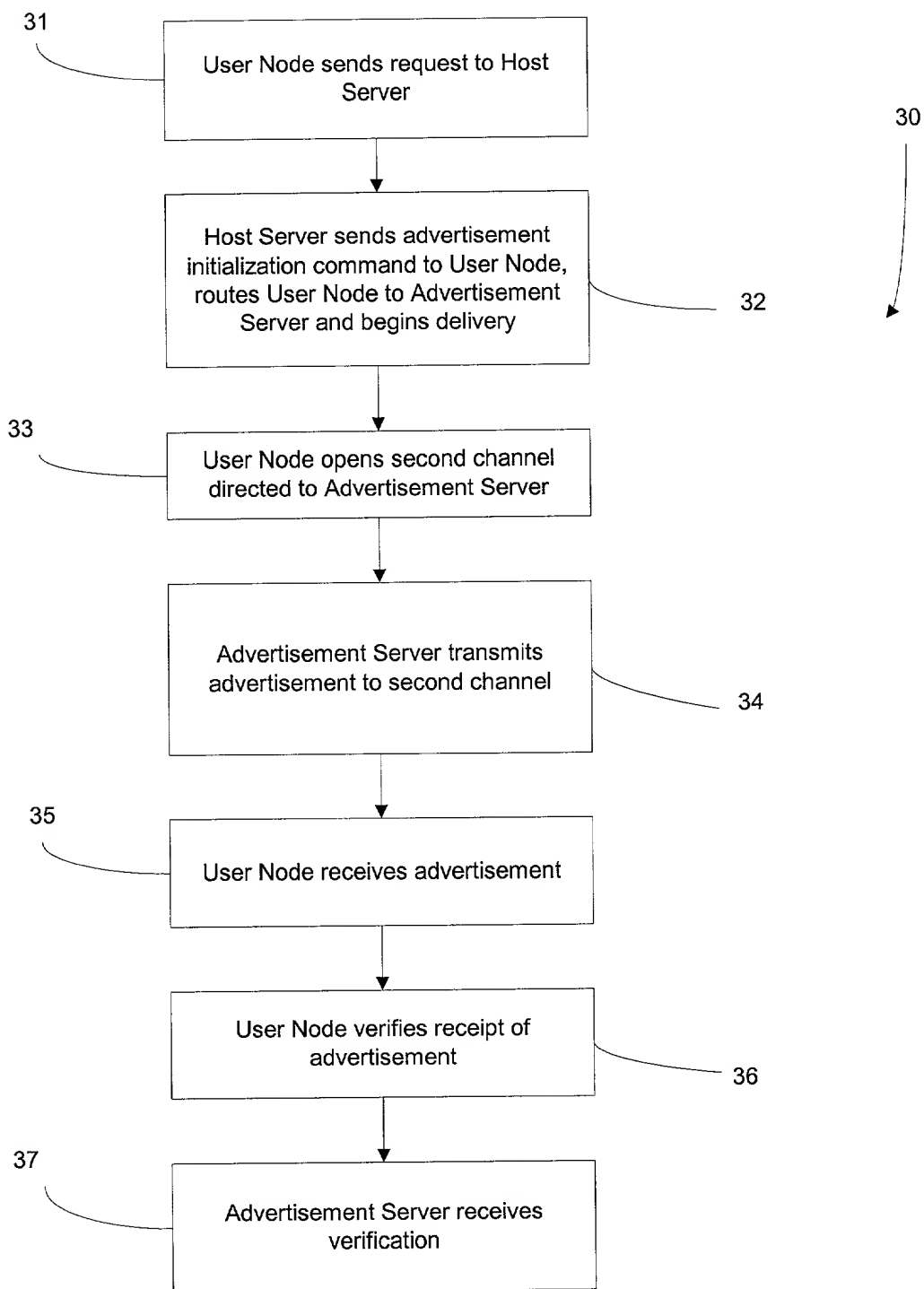


Figure 2

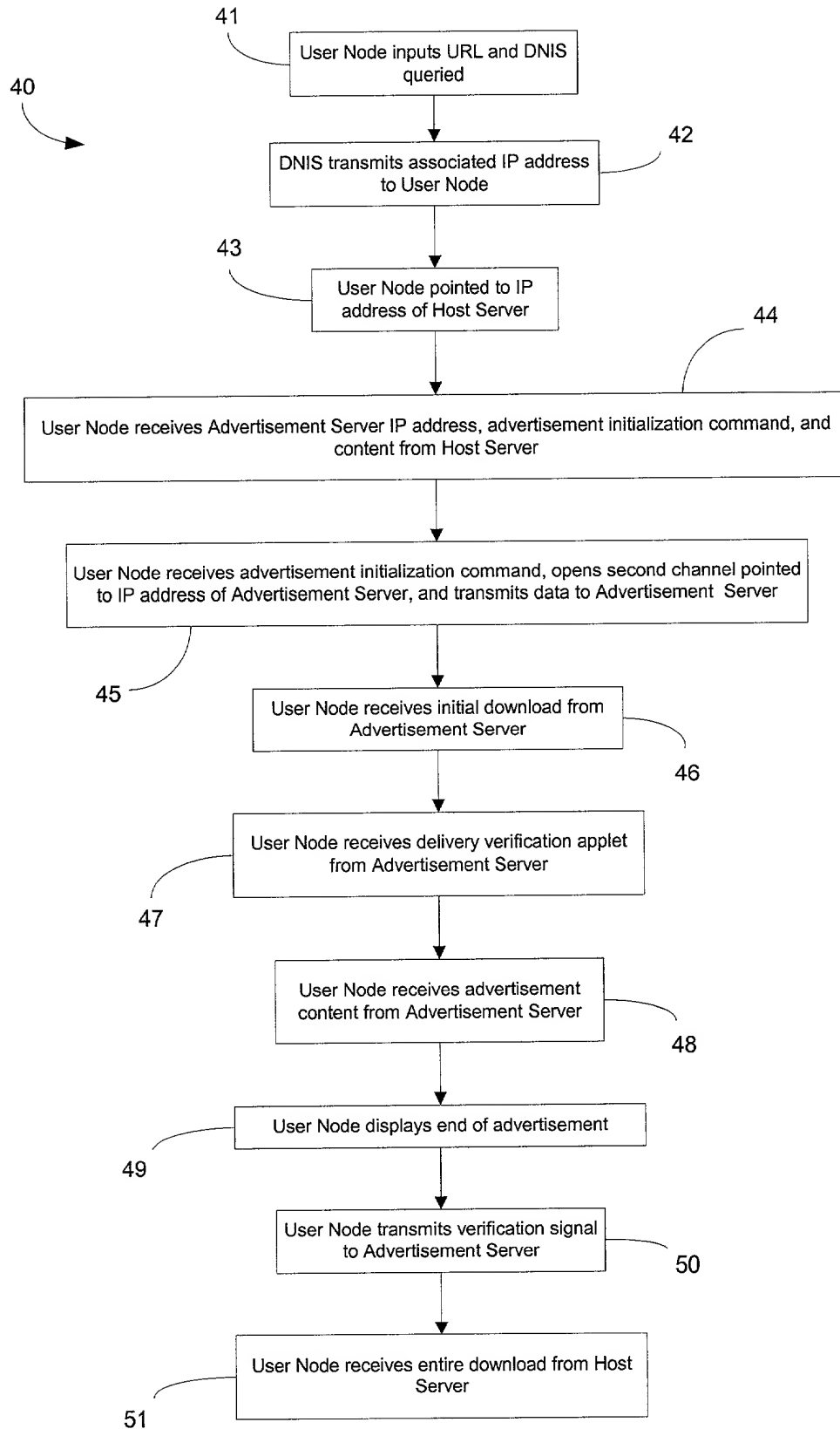


Figure 3

0976402.070601
T05040" 20499Z60

60

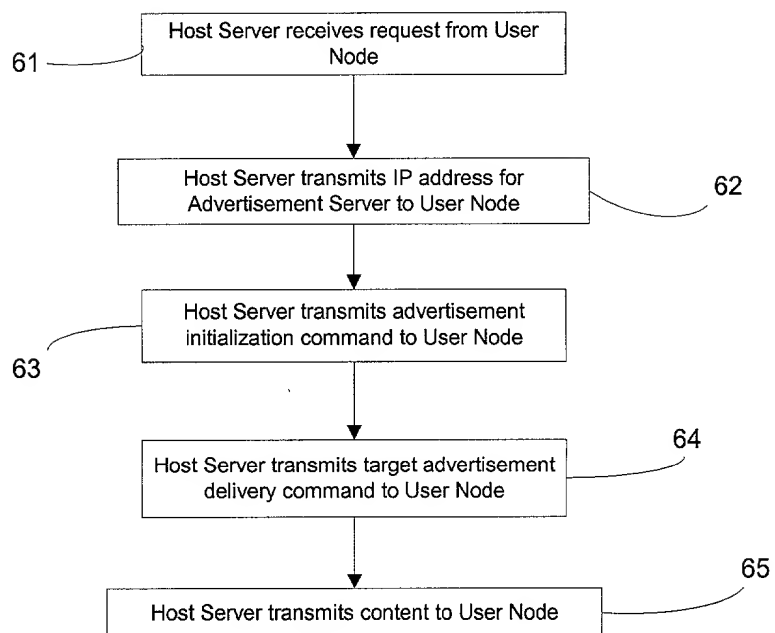


Figure 4

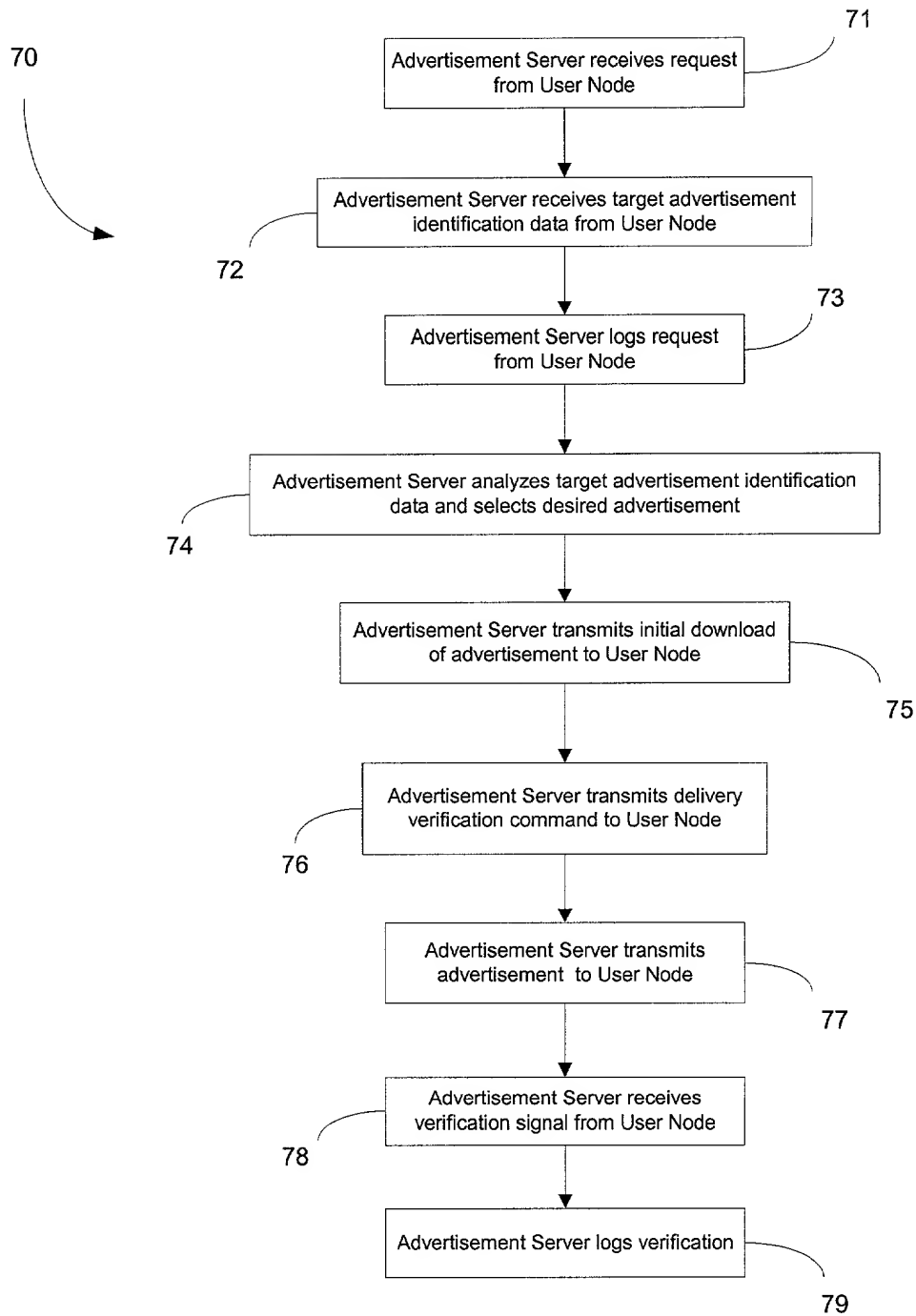


Figure 5

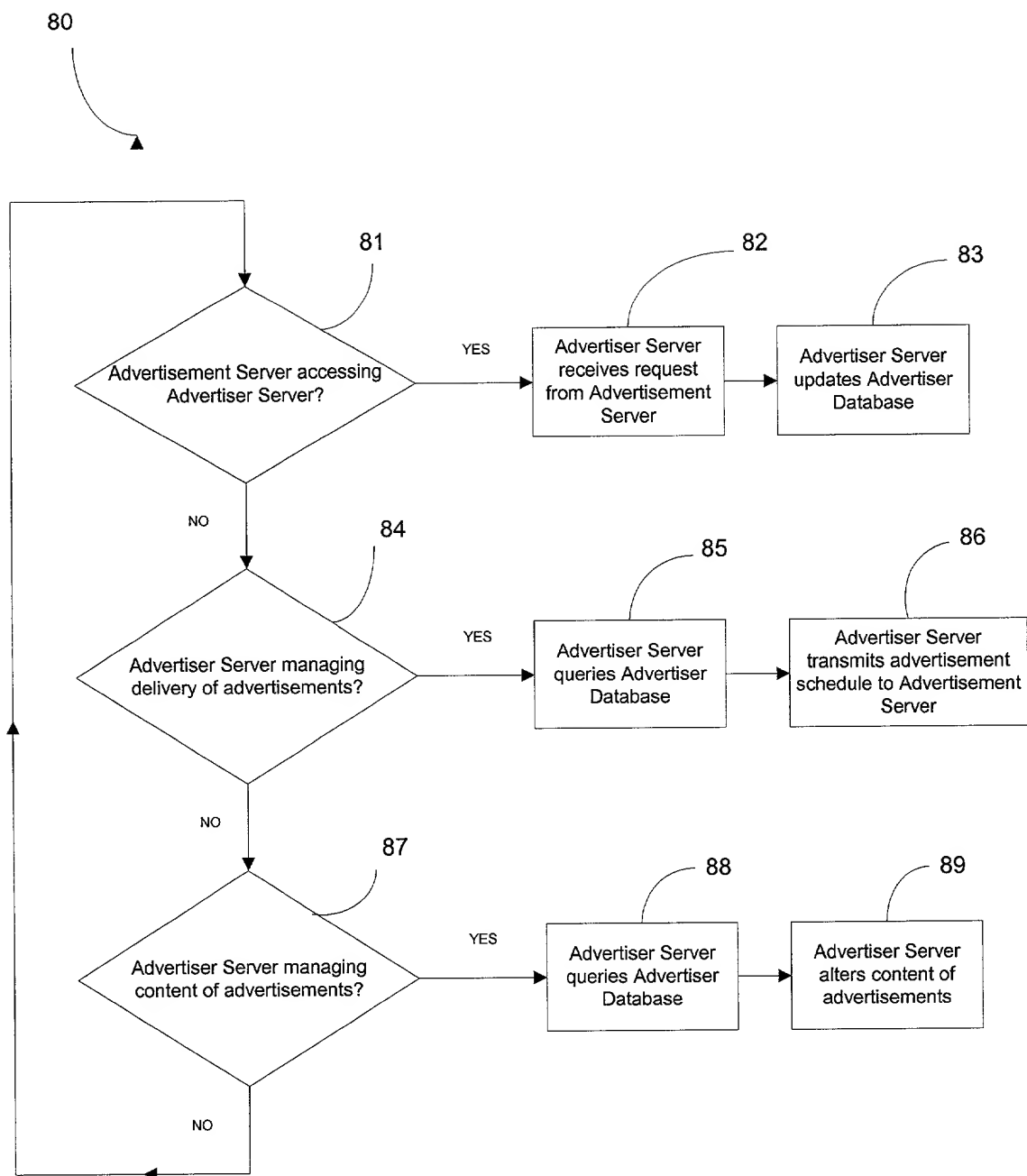


Figure 6

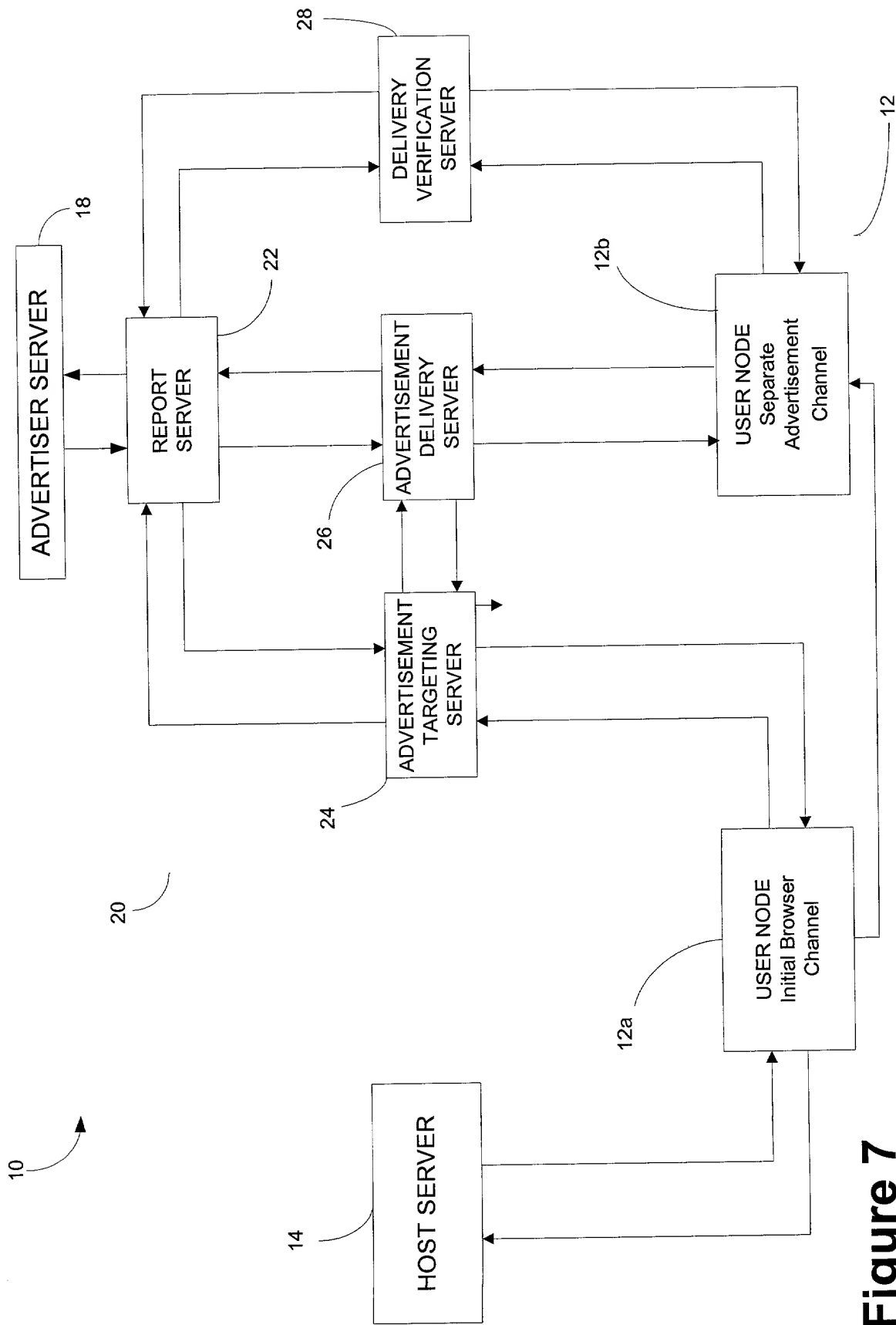


Figure 7

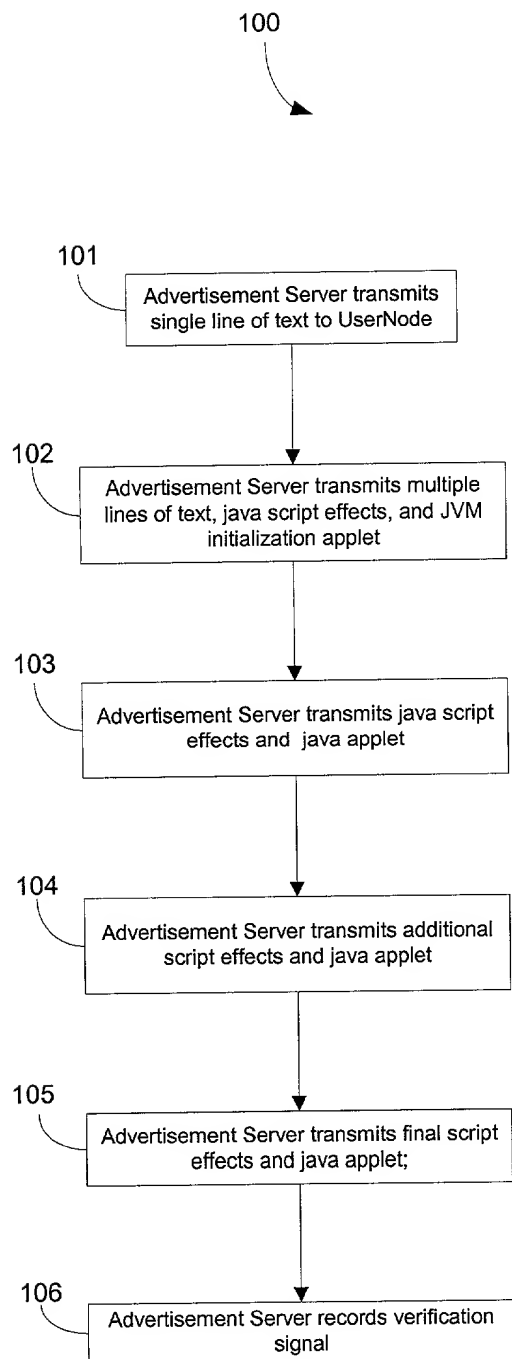


Figure 8a

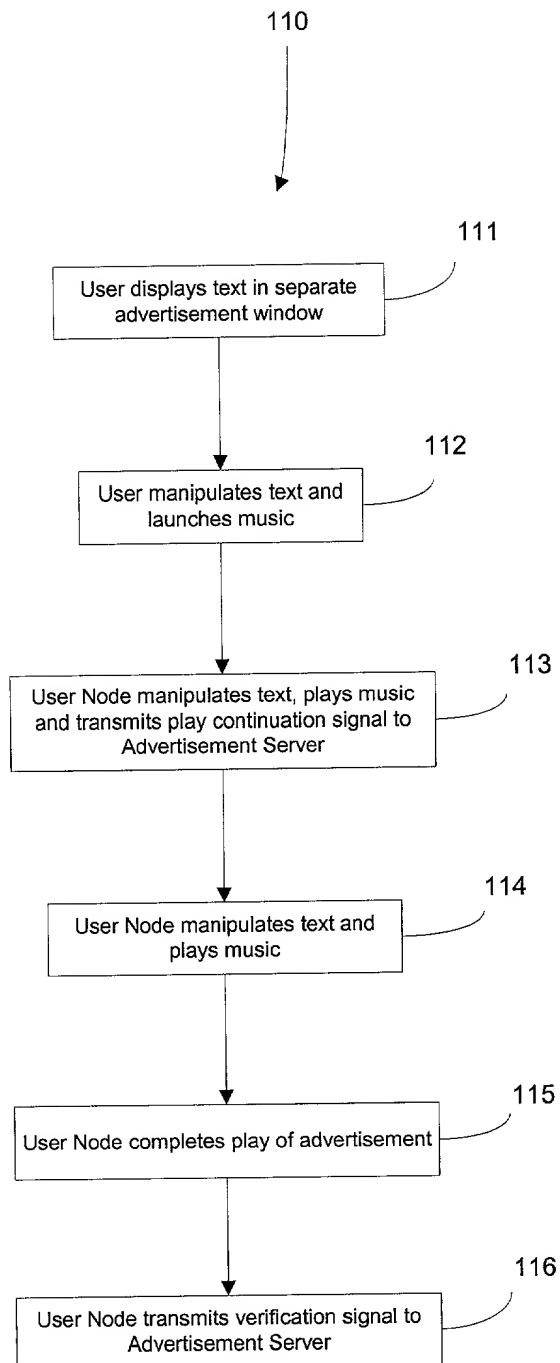


Figure 8b

09766402.070601

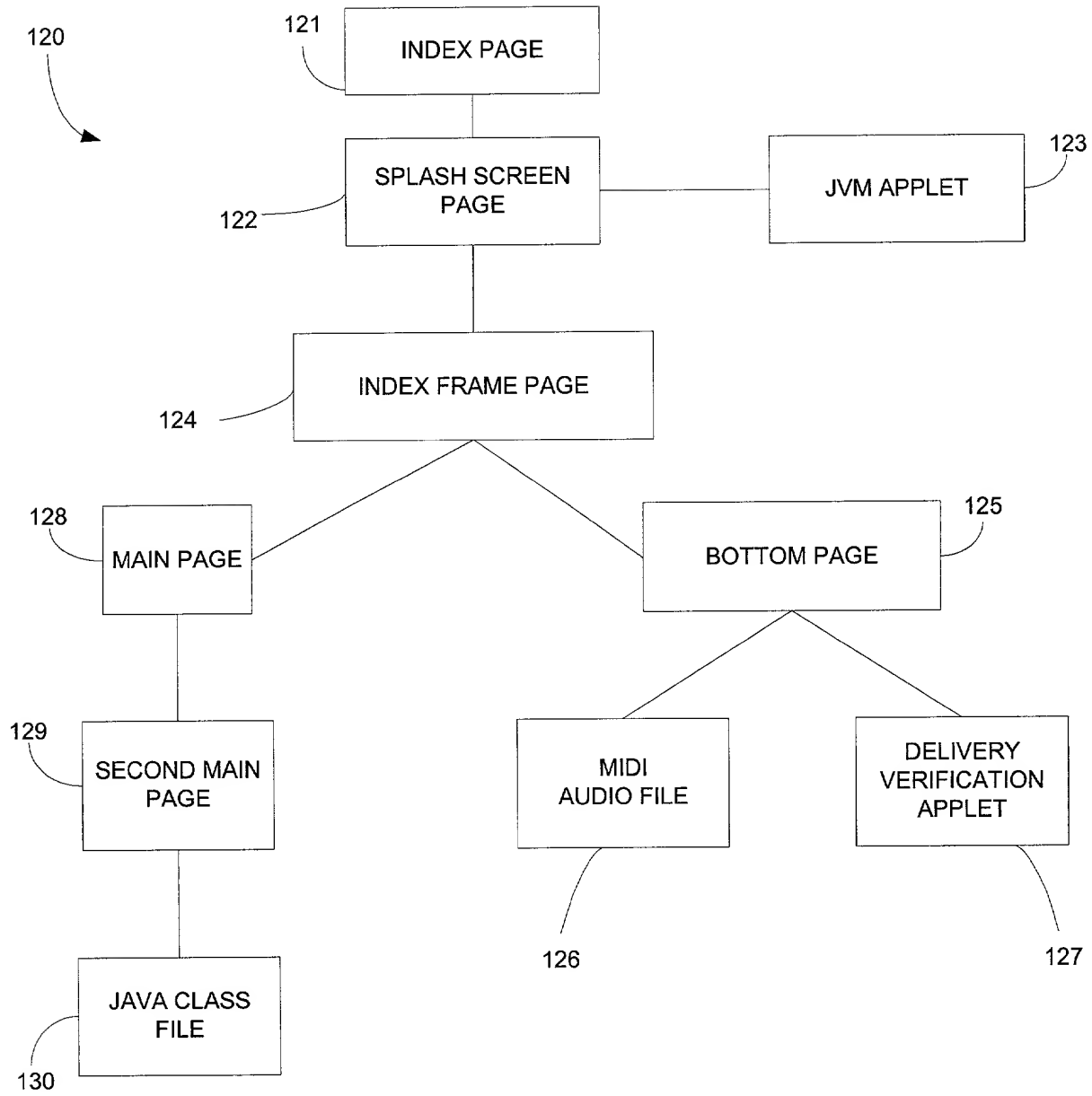


Figure 9


```

<head>
<meta http-equiv="REFRESH" content="0;URL=indexframe.html">
<title>Chevrolet Silverado</title>
</head>
<body bgcolor=#000000>

<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>

<center>
<font size="7" color="#FF6600" face="Times New Roman, Times, serif">
LIKE A ROCK
</font>

<applet code=jvm.class width=0 height=0></applet>

</center>

```

Figure 11

```

import java.applet.*;
public class jvm extends Applet {
public void init() { }
}

```

Figure 12

```

<title>Chevrolet Silverado</title>

<frameset rows="*,1" border=0>
<frame name=main src=main.html noresize>
<frame name=bottom src=bottom.html noresize>
</frameset>

```

Figure 13

```
<body bgcolor=#000000 bgsound=LIKEROK2.mid>
<embed src=likerok2.mid autostart=true width=0 height=0>
```

Figure 14

```
<meta http-equiv="REFRESH" content="2;URL=main2.html">

<body bgcolor=#000000>

<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>

<center>

<b>
<font size="7" color="#FFFFFF">
CHEVY <i>TRUCKS</i></font>
</b>
</font>

</center>
```

Figure 15

```
<head>
<script>
function closeSelf() {
  parent.close();
}

function closer() {
  setTimeout('closeSelf()',35000);
}
</script>

</head>
<body bgcolor=#000000 onload="closer();">

<p>&nbsp;</p>
<br>
<center>
<applet code=template.class width=640 height=480>
</applet>
</center>
```

Figure 16

```

import java.awt.*;
import java.applet.*;

public class tlayer extends Canvas {

    String str;
    path x,y;
    path r,g,b;
    path s;
    path z;
    path spacing;
    boolean visible,visibleAtEnd;
    int steps;
    int ttype;

    FontMetrics fm;

    public tlayer(String s) {

        str = s;
        visible = true;
        visibleAtEnd = true;

        ttype=Font.PLAIN;

    } // tlayer

```

Figure 17a

```

public void steps(int steps) {
    this.steps = steps;
}

public void position(int x1,int y1,int x2,int y2,int steps) {

    x=new path(x1,x2,steps,0);
    y=new path(y1,y2,steps,0);

} // position

public void font(int ttype) {
    this.ttype=ttype;
}

public void spacing(int x1,int x2,int steps) {

    spacing=new path(x1,x2,steps,0);

} // spacing

public void color(int r1,int g1,int b1,int r2,int g2,int b2,int steps) {
    r=new path(r1,r2,steps,0);
    g=new path(g1,g2,steps,0);
    b=new path(b1,b2,steps,0);
} // color

public void size(int s1,int s2,int steps) {
    s=new path(s1,s2,steps,0);
}

public void zindex(int z1,int z2,int steps) {
    z=new path(z1,z2,steps,0);
}

```

Figure 17b

```

public void setVisible(boolean v,boolean vAtEnd) {
    visible = v;
    visibleAtEnd = vAtEnd;
}

public void paintOffscreen(Graphics graphics) {

    // different postions,color,sizes,and whether visible

    if(visible) {

        Color c=new Color(r.getX(),g.getX(),b.getX());
        graphics.setColor(c);

        Font f=new Font("TimesRoman",ttype,s.getX());
        graphics.setFont(f);

        FontMetrics fm=getFontMetrics(f);

        int cx=x.getX();
        int cy=y.getX();
        for(int i=0;i<str.length();i++) {
            graphics.drawString(""+str.charAt(i),cx,cy);
            cx+=fm.charWidth(str.charAt(i))+spacing.getX();
        }
        // graphics.drawString(str,x.getX(),y.getX());

        // System.out.println("x1 = " + x.getX() + " y1 = " + y.getX() + " for " +
c);

    } // visible

} // paintOffscreen

```

Figure 17c


```
public void incTimer() {  
    if(steps>0) {  
        x.incStep();  
        y.incStep();  
        r.incStep();  
        g.incStep();  
        b.incStep();  
        s.incStep();  
        z.incStep();  
        spacing.incStep();  
  
        steps--;  
  
        if(steps==0 && !visibleAtEnd)  
            visible=false;  
    }  
} // incTimer  
  
} // tlayer
```

Figure 17d

```

import java.awt.*;
import java.applet.*;

public class ilayer extends Canvas {

    Image im;
    path x,y;
    path sx,sy;
    path z;
    boolean visible,visibleAtEnd;
    int steps;

    public ilayer(Image im) {

        this.im=im;

        visible = true;
        visibleAtEnd = true;

    } // tlayer

    public int getX() {
        return x.getX();
    }
    public int getY() {
        return y.getX();
    }
    public int getSX() {
        return sx.getX();
    }
    public int getSY() {
        return sy.getX();
    }
}

```

Figure 18a

```

public boolean getVisible() {
    return visible;
}

public void steps(int steps) {
    this.steps = steps;
}

public void position(int x1,int y1,int x2,int y2,int steps) {

    x=new path(x1,x2,steps,0);
    y=new path(y1,y2,steps,0);

} // position

public void size(int x1,int y1,int x2,int y2,int steps) {

    sx=new path(x1,x2,steps,0);
    sy=new path(y1,y2,steps,0);

} // position

public void zindex(int z1,int z2,int steps) {
    z=new path(z1,z2,steps,0);
}

public void setVisible(boolean v,boolean vAtEnd) {
    visible = v;
    visibleAtEnd = vAtEnd;
}

```

Figure 18b

```

public void paintOffscreen(Graphics graphics) {

    // different postions,color,sizes,and whether visible

    if(visible) {

        graphics.drawImage(im,x.getX(),y.getX(),sx.getX(),sy.getX(),this);

        //      System.out.println("x1 = " + x.getX() + " y1 = " + y.getX());

    } // visible

} // paintOffscreen

public void incTimer() {

    if(steps>0) {

        x.incStep();
        y.incStep();
        sx.incStep();
        sy.incStep();

        z.incStep();

        steps--;

        if(steps==0 && !visibleAtEnd)
            visible=false;

    }

} // incTimer

} // ilayer

```

Figure 18c

```

public class path {

    double x1,x2;
    double xstep;
    int eq,steps;

    public path(double x1,double x2,int steps,int eq) {

        this.x1=x1;
        this.x2=x2;

        this.steps=steps;
        this.eq=eq;

        xstep=(x2-x1)/(double) steps;
        // System.out.println("**** init x1 = " + x1 + " x2 = " + x2 + " step = " + xstep);

    } // path constructor

    public void incStep() {

        x1=x1+xstep;

    } // incStep

    public int getX() {
        return (int) x1;
    }

} // path class

```

Figure 19

```
/*  
    Java DVT Client --> connecting to --> Multi-threaded DVT C Server
```

```
*/
```

```
import java.awt.*;  
import java.io.*;  
import java.net.*;  
import java.lang.*;  
import java.applet.*;
```

```
public class dtvclient extends Applet implements Runnable {
```

```
    Socket clientSocket=null;  
    DataInputStream dis=null;  
    OutputStream os=null;  
    String host;  
    int port;  
    int delay;  
    boolean isRunning=true;  
    int messages=0;
```

```
    Thread threadRef=null;
```

```
public void init() {
```

```
    System.out.println("starting constructor");  
    host="38.202.155.30";  
    port=2048;  
    delay=1000;
```

```
    System.out.println("Done with constructor");
```

```
    // create socket communications
```

```
    System.out.println("Attempting to connect to port "+host+": "+port+"\n");
```

Figure 20a

```

try {
    clientSocket=new Socket(host,port);
} catch(Exception makingsocket) {
    System.out.println("Error connecting to " + host + " at port " + port);
    return;
}

System.out.println("Made Connection...");

try {
    dis=new DataInputStream(clientSocket.getInputStream());
    os=clientSocket.getOutputStream();

    } catch(UnknownHostException e) {
        System.out.println("Unknown Host exeception getting socket streams!!!");
    } catch(IOException e) {
        System.out.println("IO exeception getting socket streams!!!");
    }

System.out.println("Made input/output connections");

threadRef = new Thread(this);
threadRef.start();

} // constructor

```

Figure 20b

```

public synchronized void start() {

    if(threadRef==null) {
        System.out.println("Null threadRef in start()");
        threadRef = new Thread(this);
        threadRef.start();
    }

} // start


public void stop() {

    System.out.println("Stopping...");
    if(threadRef!=null) {

        threadRef.stop();

        threadRef=null;
    }
} // stop


public void destroy() {

    System.out.println("Destroying...");
    if(threadRef!=null) {

        threadRef.stop();

        threadRef=null;
    }
} // destroy

```

Figure 20c


```

public void run() {

    System.out.println("Starting run..");
    while(isRunning && messages<3) {

        SendAndReceive();
        messages++;
        System.out.println("Going to sleep for 1 second");
        try {
            threadRef.sleep(delay);
        } catch(Exception e) {
            System.out.println("Error in sleep");
        }

    } // while isrunning

    try {
        System.out.println("Closing socket...");
        clientSocket.close();
        stop();
        destroy();
    } catch(Exception e) {
        System.out.println("Error Closing socket");
    }

} // run

```

```

void SendAndReceive() {

    byte bbuf[]=new byte[256];
    String str;

    try {

```

Figure 20d

```

System.out.println("command being sent.");
str="message "+messages;
for(int i=0;i<str.length();i++)
    bbuf[i]=(byte) str.charAt(i);
bbuf[str.length()]='\0';
os.write(bbuf,0,str.length());
os.flush();
System.out.println("Command written to server");

```

```

// read string back
str= dis.readLine();

```

```

System.out.println("Got: " + str);

```

```

if(str.length()==0) {
    System.out.println("ERROR receiving from " + host + ":" + port);
    clientSocket.close();
    return;
}

```

```

} catch(Exception e) {
    System.out.println("Exception during send/receive");
}

```

```

} // SendAndReceive

```

```

} // dvtclient

```

Figure 20e

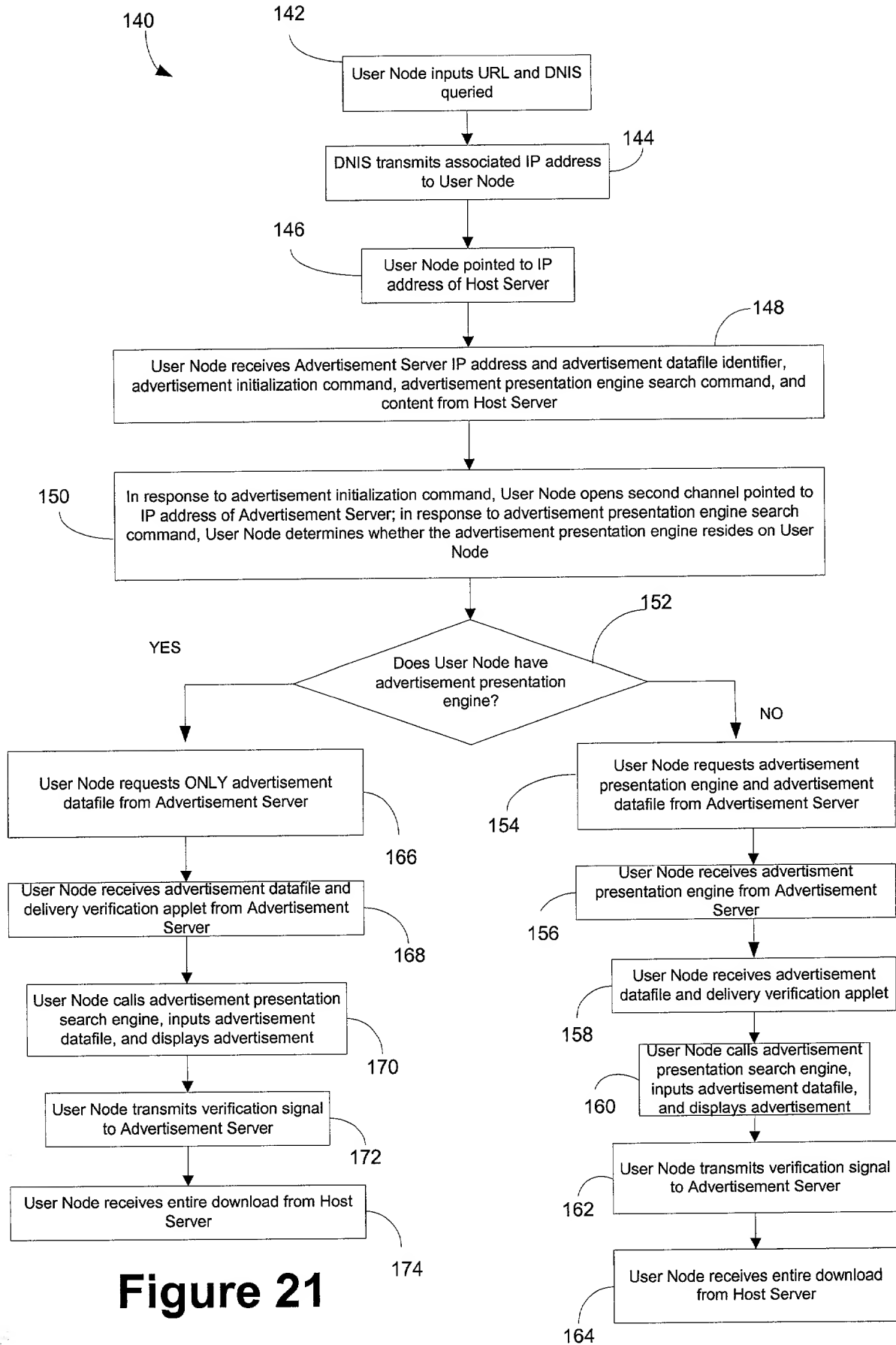


Figure 21

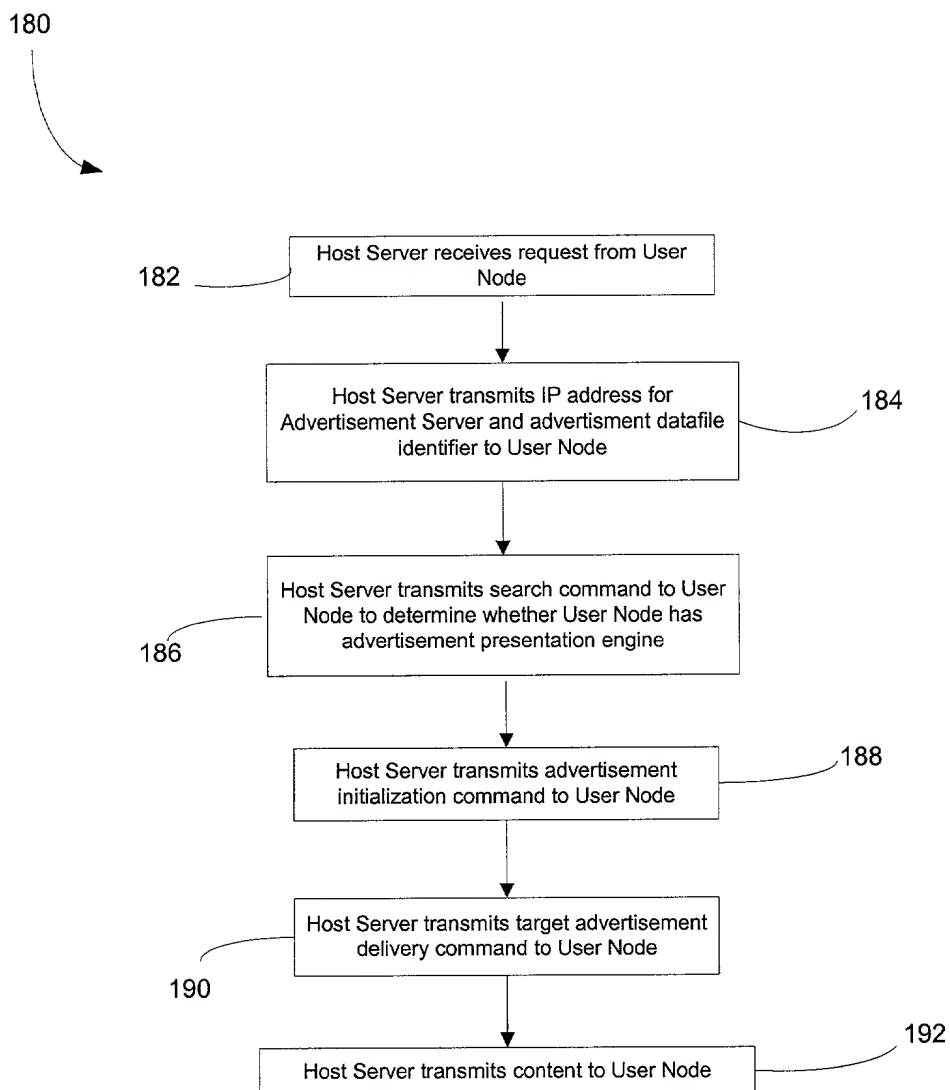


Figure 22

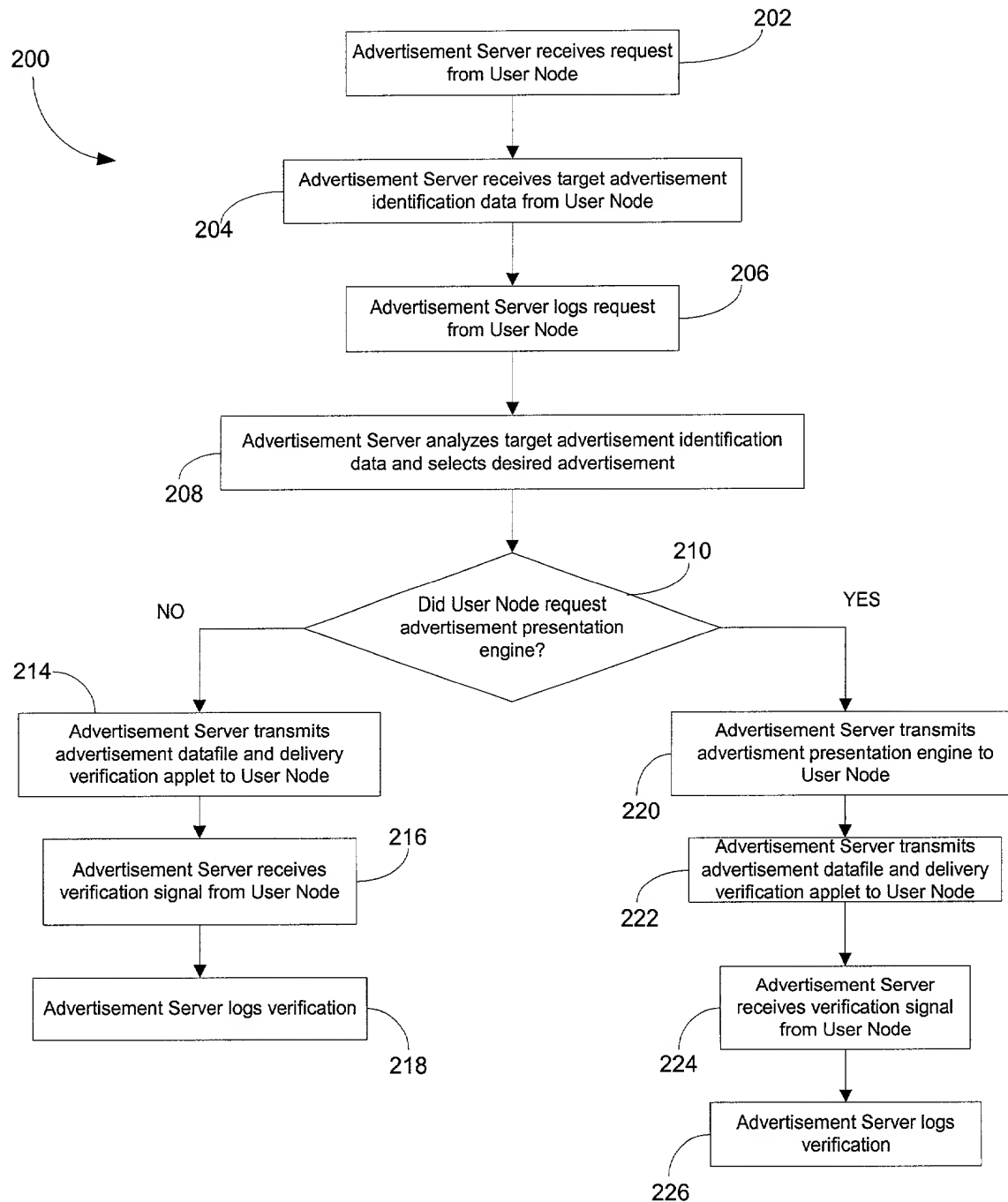


Figure 23

09763402.070601

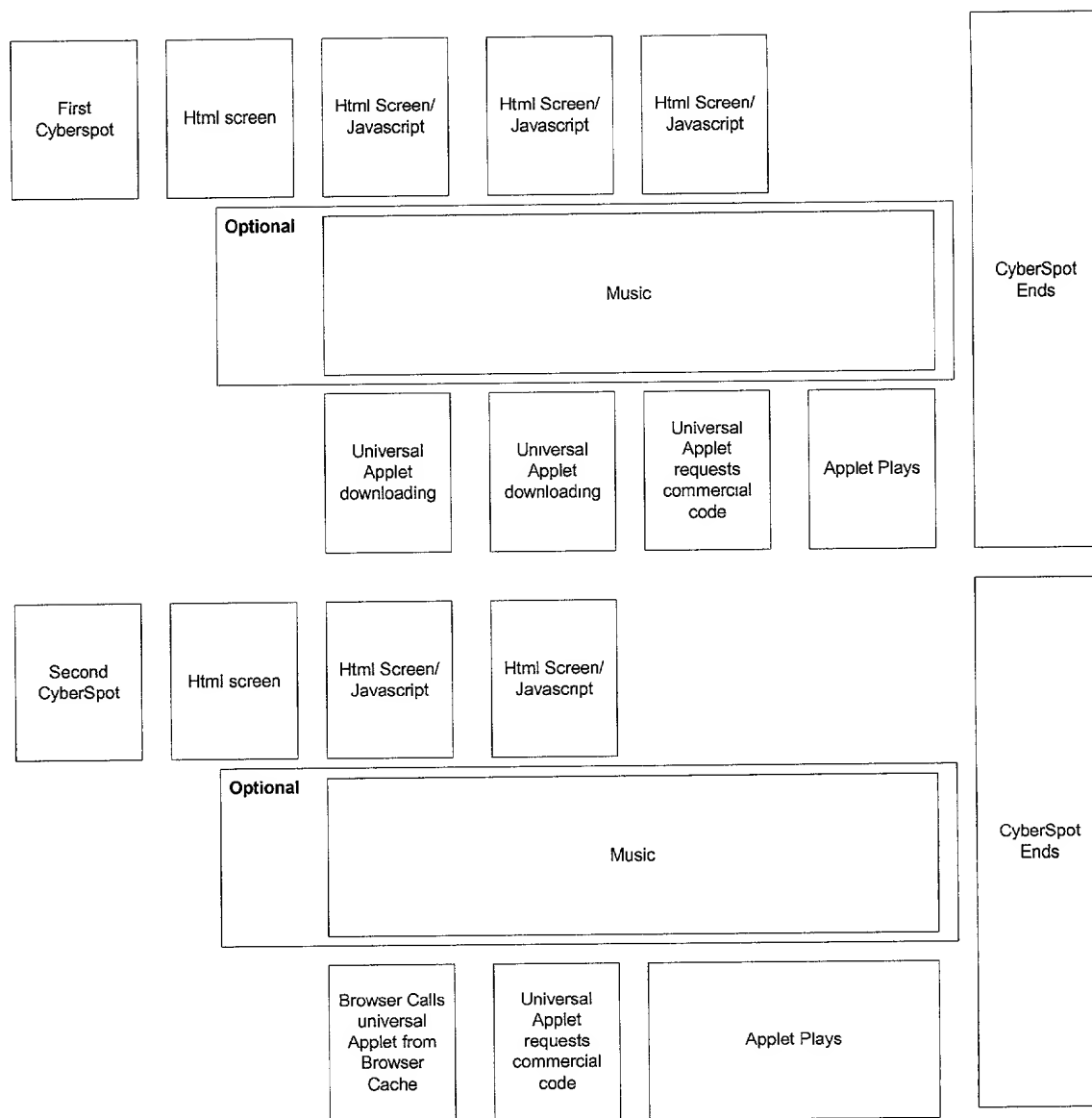


Figure 24(a)

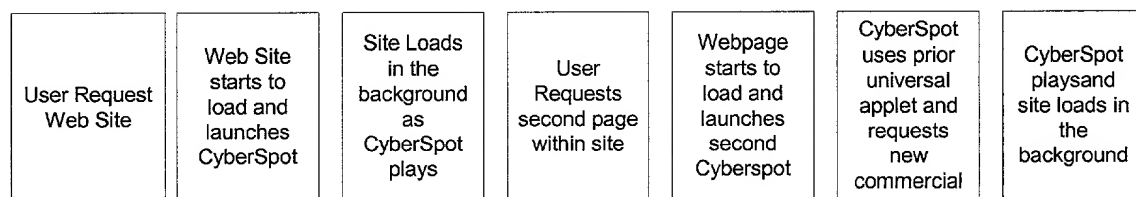


Figure 24(b)

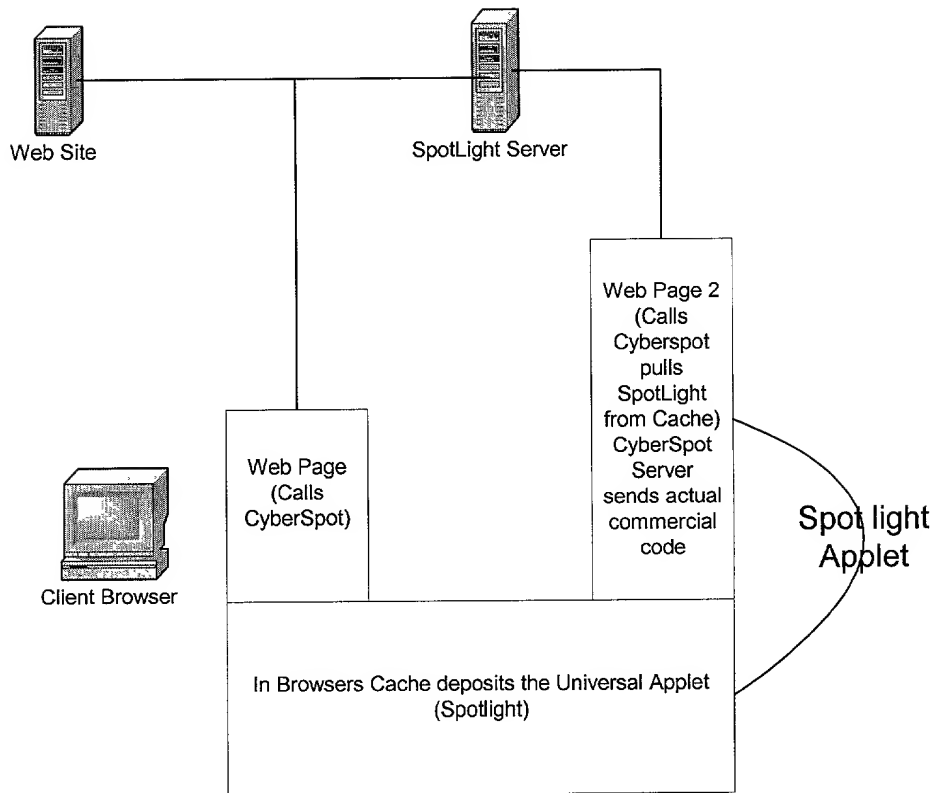


Figure 25